# METHOD, SYSTEM, AND STORAGE MEDIUM FOR MANAGING COMPUTER PROCESSING FUNCTIONS

## CROSS-REFERENCE TO RELATED APPLICATION

**[0001]** This application is related to and incorporates integrating processing functions as provided in U.S. patent application entitled "METHOD, SYSTEM, AND STORAGE MEDIUM FOR SEARCHING MULTIPLE QUEUES FOR PRIORITIZED WORK ELEMENTS", attorney docket number POU920040047US1, filed concurrently with this application.

## BACKGROUND OF THE INVENTION

**[0002]** The present invention relates generally to computer processors, and more particularly, to a method, system, and storage medium for managing computer processor activities in a multi-processor computer environment

**[0003]** In a multi-processing computer system, multiple CPUs share various processing tasks requested by the operating system and applications executing on the system. Not all CPUs are equal in their ability to perform these tasks. Some processors are, by design, faster or more functional than others. Clearly, the costs of these processors increase in proportion to their speed and functionality. The processing needs and requirements of a system owner may vary over time or depending upon the types of functions to be performed on the system.

**[0004]** In contemporary computer networks, it is a common practice to interconnect multiple computing systems, typically called servers, in a manner that segregates each of the interconnected servers into separate usage classes based on the functions that each of these servers is configured to execute. For example, one or more of the interconnected servers may be assigned the task of executing application programs

and others may be assigned the task of executing database programs that access and manage one or more of the network's associated databases. Networks that are structured in this manner are typically described as multiple-tier or multi-tier server networks.

**[0005]** The structuring of multi-tier networks and the assignment of work to each of the servers within the network is typically determined by a number of factors such as server reliability, server scalability, server functional capabilities, server cost, etc. Typically the computing functions that require the highest degrees of reliability, availability, scalability, and performance are assigned to the database servers that control the management and access of the network user's data. In other words, the core of many multi-tier computing networks, in terms of user data availability and reliability, lies at the data servers which are assigned the task of accessing, maintaining, and keeping up-to-date a centralized, integrated database upon which many companies rely for maintaining and growing their associated business. As such, the database servers are often the most reliable and functionally robust computing elements within the network and because of this are also quite often the most expensive servers within the network in terms of total "cost of ownership".

**[0006]** Current and prior attempts to simplify and reduce the cost of multi-tier server networks by eliminating or minimizing the number of application servers and integrating the application programming functions onto combined application/database servers has had marginal success. This has typically been true because of the increased cost of ownership associated with the more robust database servers. Correspondingly, the proliferation of "dedicated" application servers within such networks is the current reality for many of today's network environments. While the initial cost to purchase and deploy additional application servers may be less than contemporary costs for integrating the application server's programming functions onto a large multi-function database server, such proliferation of these distributed application servers can typically increase the overall complexity and reduce the reliability of the network, and over time, can result in a

greater total cost of ownership for the network in order to maintain the increased total number of server computing elements within the network.

**[0007]** Another factor which has resulted in significant increases in the number of dedicated application servers in contemporary multi-tier networks is the programming technologies currently being deployed in order to implement timely and cost effective applications. This is especially true for applications designed to support the evolving strategic e-business and on-demand business models. Programming languages such as JAVA and data structures such as XML, both of which play a major role in developing new strategic applications, provide significant advantages in terms of programming development costs and time to market, due to high level, application based, programming models they present to the program developer. Such high level programming languages are typically computer architecture agnostic in order to render the application programs operable on different hardware architecture server platforms. However, this "application can operate anywhere" design point has also resulted in significantly increasing the processing requirements of such applications in terms of actual processor instructions that must be executed in order to support a given application program. For example, new JAVA applications are emerging that can require the execution of as many as 75 million hardware instructions in order to process a single application transaction. Consequently, such applications are typically not cost effective when executed on more expensive database server platforms. The cost per unit of work is simply too great to deploy such applications on the typically more expensive, but highly reliable, database servers.

**[0008]** What is needed therefore is a way to consolidate the processing functions performed in a multi-processor computer environment.

## SUMMARY OF THE INVENTION

**[0009]** The above stated disadvantages are overcome or alleviated by a method, system, and storage medium for managing computer processing functions in a multi-processor computer environment. The method includes invoking a switch-to service by

standard code. The standard code is running on a standard logical processor and is executing a task. The switch-to service checks to see if an assist logical processor is online, and if it finds one, the switch-to service updates an integrated assist field of a work element block associated with the task and assigns a queue to the work element block. The task is dispatched in accordance with business rules identified in a system control block associated with the task and is executed by assist code running on the assist logical processor.

## BRIEF DESCRIPTION OF THE DRAWINGS

**[0010]** Referring now to the drawings wherein like elements are numbered alike in the several FIGURES:

**[0011]** FIG. 1 is a block diagram of a multi-processor computer system upon which the integrated assist system may be implemented in exemplary embodiments;

**[0012]** FIG. 2 is a flowchart describing a process for implementing the integrated assist system in exemplary embodiments;

**[0013]** FIG. 3 is a sample work element block utilized by the integrated assist system in exemplary embodiments;

**[0014]** FIG. 4 illustrates system work queues that hold pending dispatch to a processor in exemplary embodiments; and

**[0015]** FIG. 5 illustrates a sample system control structure utilized by the integrated assist system in exemplary embodiments.

## DETAILED DESCRIPTION OF THE INVENTION

**[0016]** The integrated assist system of the invention simplifies the deployment of computer processing systems such as single, stand-alone computers as well as multi-tiered network computer systems and improves their processing efficiency, overall

reliability, and total cost by consolidating the processing functions to meet the needs of the system user. In a stand-alone computer processing environment, multiple CPUs may be classified as standard and assisted by the integrated assist system. These classifications are flexible and may be tailored to the needs of the system owner. In a multi-tier network system, the integrated assist system integrates the processing functions that are typically performed by less costly and functionally less rich systems, such as application servers, with those functions performed by more costly and more robust systems, such as database servers, onto a single integrated multi-functional platform. The integrated assist system utilizes an integrated assist processor that is incorporated into the existing platform to form a heterogeneous processor complex capable of executing both of the above processing systems (e.g., application and database programming) in a manner that is both more efficient and reliable than 3-tier networks and which can compete with such networks on a cost of ownership basis.

**[0017]** Referring now to FIG. 1, a block diagram of a system 100 upon which the integrated assist system may be implemented is described. System 100 may comprise a single, stand-alone computer system utilizing multiple processors or may be a collection of several processors and applications found in a large multi-tier network. In a multi-tiered network, for example, the processors are typically grouped by functions or application types.

**[0018]** System 100 includes a logical partition (LPAR) container 102. LPAR container 102 is a repository that organizes the physical processors of pools 114 and 116 into virtual groupings of standard logical processors 104-106 and integrated assist logical processors 108-110 as described further herein. These groupings are transparent to the operating system and application programs/processes executing on each processor. It will be understood by those skilled in the art that any number of logical and physical processors may be employed by the computer system in order to realize the advantages of the invention.

[0019] Standard logical processors 104 and 106 refer to virtual processors that result from the partitioning of actual standard physical processors of pool 114 and are logically addressable by both logical partition hypervisor 112 and the programs/processes executing on them, as well as the integrated assist processors 108 and 110, as if they were actual physical assist processors. Standard logical processors 104 and 106, as well as standard physical processors in pool 114, refer to computer processors (virtual and physical) that are typically assigned specific tasks, processes, or work in a computer processing environment. Standard logical/physical processors (104, 106, 114) differ from integrated assist logical/physical processors (108, 110, 116) in that integrated assist logical/physical processors are assigned to 'help' standard logical/physical processors perform its assigned tasks. Work that is restricted to standard processors (referred to herein as 'standard work') may be defined as work that is optimally performed on a standard processor due to cost constraints, the critical nature of the data to be processed, logistics relating to memory constraints associated with physical processors, system or user needs, etc.). All other work is referred to herein as assisted work. Essentially, a system administrator, operating system provider, or other entity may define what types of work should be classified as assisted work (e.g., non-critical, non-priority, etc.) and what types of work should be classified as standard work. Further, a system administrator, operating system provider, etc., may define to what extent (and under what conditions) assisted work may be performed on a standard logical processor.

[0020] Integrated assist logical processors 108 and 110 comprise virtual processors that result from the partitioning of actual integrated assist physical processors of pool 116. Similar to standard logical processors 104 and 106, integrated assist logical processors 108 and 110 are logically addressable by both the logical partition hypervisor 112 and the programs/processes executing on them.

[0021] As shown in system 100, integrated assist logical processor 108 and standard logical processor 106 are assigned to a given logical partition (102) and are both accessible by the programs/processes residing in the partition memory or central storage.

Integrated assist logical processors 108, 110 attach to and utilize the same memory interfaces and functions as that of the standard logical processors 104, 106 to which they are configured via the integrated assist system. This configuration enables seamless and transparent inter-communication between the processors 104-110. It will be understood that the configuration between a standard logical processor and an integrated assist logical processor is not limited to a one-to-one mapping. In other words, a joint configuration may be possible between an integrated assist logical processor and multiple standard logical processors with respect to a given logical partition. Thus, the depiction as shown in FIG. 1 is shown for illustrative purposes and is not meant to be construed as limiting in scope. Standard physical processors in pool 114 and integrated assist physical processors in pool 116 are depicted as "shared" processors in that they may be shared by other LPARs in addition to LPAR 102. It will be understood that these processors are not limited to shared processors but may be dedicated to a specific logical partition if desired.

**[0022]** A single operating system instance 119 is associated with both of standard logical processors 104, 106 and integrated assist logical processors 108, 110. The operating system 119 executing within LPAR 102 may be a commercial enterprise operating system application such as IBM's z/OS™. The integrated assist system schedules or allocates the logical processors 104-110 to the relevant application programs/processes via operating system 119 (e.g., standard application code 120, 124 and assisted code 122 allocated to standard logical processor 106 and integrated assist logical processor 108.

**[0023]** A task 118 is shown executing within logical partition 102. Standard logical processor 106 and integrated assist logical processor 108 may utilize both standard code 120, 124 and assisted code 122 to perform task 118. Standard code and assist code may be defined in accordance with the business needs of an enterprise utilizing the integrated assist system. Determining what code should be run on a standard processor and what code should be run on an assist processor might be a product of the functionality requirements, cost considerations, processor speeds, business goals, etc. of

the enterprise. For example, A manufacturer may want to charge more for an assist processing engine and limit the code that will run on it to that which will enable the assist processing engine to perform faster. Another manufacturer may wish to charge lower prices for an assist processing engine and run code that is known to be somewhat slower than a standard processor. Standard code 120, 124 and assisted code 122 utilize a "switch-to" service and a "switch-from" service as described further herein.

**[0024]** Logical partition hypervisor 112 refers to an operating system that maps standard logical processors to standard physical processors and further maps integrated assist logical processors to integrated assist physical processors. Hypervisor 112 allocates the physical processors of pool 114 and 116 to logical partitions such as LPAR 102. Resources within LPAR 102 such as memory size and processor time are managed by hypervisor 112. Hypervisor 112 ensures that processors 104-110 are individually addressable and dispatchable as separate physical processors as if they were jointly configured standard physical processors. Hypervisor 112 dispatches integrated assist logical processors (e.g., 108, 110) on integrated assist physical processors in pool 116 and dispatches standard logical processors (e.g., 104, 106) on associated standard physical processors in pool 114.

**[0025]** Turning now to FIG. 2, a process for implementing the integrated assist system will now be described. The integrated assist system comprises a 'switch-to' service component and a dispatch component. Utilizing a work element block (shown in FIG. 3), the switch-to service component performs authorization of the calling applications/processes such as standard code 120 and verifies availability of integrated assist physical processors (e.g., processors in pool 116). The dispatch component is managed by operating system 119 and differentiates between types of work (e.g., standard work, assisted work) and includes rules for determining when and if assisted work will run on a standard processor. The process steps recited in FIG. 2 presuppose that the integrated assist code 122 executing on processor 108 is a JAVA application and that the JAVA application is initiating a call to a database management system program

(i.e., standard code 124) executing on standard logical processor 106. Both standard logical processor 106 and integrated assist logical processor 108 are executing in a Java Virtual Machine (JVM) environment for enabling Java code to be executed in varying software/hardware environments. It will be understood by those skilled in the art, that other assist applications may be used in lieu of the Java application such as, for example, an XML program.

**[0026]** At step 202, the standard code 120 calls for a switch-to service to an assist processor for a given task 118. The switch-to service may be a routine within the operating system 119 as specified by a system administrator, a hardware/software policy, or other means. As described above, there are various reasons why a standard processor may wish to switch-to an assist processor. The switch-to service performs an authorization for the calling application at step 204. The authorization may be based upon a variety of factors such as the name of the module calling the switch-to service, the user identity of the process calling the switch service, a permission to a security product defined in an access list, an authorization flag associated with the executing process, or a combination of the above. Further, if desired, there may be no authorization specified. That is, the provider of the integrated assist processor enables the processor to be available to any application that chooses to implement the call to the switch-to service.

**[0027]** At step 206, it is determined whether the calling application is authorized to make the switch-to service call. If not, the switch-to service returns a failure code to the calling application at step 208. If the calling application is authorized at step 206, the switch-to service determines if the integrated assist processors are available. For example, an integrated assist processor may not be defined or configured to the system. Alternatively, the integrated assist processor may be configured but not online. Those integrated assist processors that are not online are deemed to be "disabled." A processor may be disabled by the integrated assist system in order to prevent unauthorized access to them when an operating system that does not recognize these integrated assist processors is invoked. Hypervisor 112 includes a definition of the LPAR container 102 which

provides information about the system configuration. Hypervisor 112 provides this configuration information to operating system 119 utilizing a READ SCP information service call. Each standard logical processor and integrated assist logical processor address includes a single byte used as a processor type code. A type code of '00'x indicates a logical standard processor and a non-zero value type code represents an integrated assist logical processor. Using this type code information, a CP_mask 506 has a bit turned on for each online standard processor and an IAP_mask 508 has a bit turned on for each online assist processor. This information is stored in a system control block 500 as shown in FIG. 5.

**[0028]** When an operating system enabled to handle integrated assist processors is invoked, it issues a SIGP instruction via the integrated assist system, which causes the integrated assist processors 108, 110 in LPAR 102 to become enabled for use. Following this SIGP instruction, integrated assist processors can be started, stopped, etc., in the same manner as any other standard logical processors in the LPAR configuration.

**[0029]** The switch-to service then determines which assist processors are online by accessing mask 508 in system control block 500. CP_MASK 506 identifies which of the online processors is a standard processor. IAP_MASK 508 identifies which of the online processors is an integrated assist processor. When the time comes for the dispatch component to take control, it checks masks 506 and 508 to see which processor it is running on.

**[0030]** If an integrated assist processor is available at step 210, the switch-to service sets the return code to success at step 212, followed by updating a work element block associated with the task to indicate that the task is eligible at step 214. This is accomplished by setting the IAP indicator field 306 of the work element block 300 of FIG. 3. The work element block 300 contains a chain pointer 302, a priority field 304, and the IAP indicator field 306. If there is only one type of integrated assist processor, then IAP indicator may be a flag. Otherwise, IAP indicator 306 may be an index into a list of IAP processor types. The work element block 300 is assigned to one of system

work queues 402 and 404 as shown in FIG. 4, depending upon the type of work required at step 216. System work of each type is maintained in separate control structures or queues, one queue for each type of work. In alternate embodiments, a single queue can be used as long as each type of work can be differentiated by looking at the individual work element. The process returns to step 202.

**[0031]** When the unit of work becomes dispatchable, the work element block 300 is placed on the appropriate queue based upon the IAP indicator field 306. The work/task is dispatched in accordance with system control structure indicators 502-510 as described further herein.

**[0032]** If no integrated assist processors are available at step 210, the switch-to service returns a warning code to the calling application at step 218 and updates work element block 300 associated with the task at step 220 to indicate that the task 118 is eligible for processing via an integrated assist processor. This is performed to allow for the possibility that an integrated assist processor will be brought online in the future. By continuing to set the IAP indicator field 306 of work element block 300, the calling application will not require restarting when an integrated assist processor is brought online. The process returns to step 202 whereby the standard code 120 continues to call the switch-to service.

**[0033]** Once the assist code 122 is finished with the data/process, it calls the 'switch-from' service which causes the standard code 124 to be resumed. The return switch-from service process does not require any authorization or availability checks.

**[0034]** As indicated above, different types of work are differentiated by the integrated assist system. Assisted work may be run on an integrated assist processor but may also be run on a standard processor. Assisted work may be further defined or classified into any number of types of assisted work (e.g., assisted-1, assisted-2, assisted-3, assisted-n, whereby assisted-n work may be run on an assist-n integrated assist

processor). For example, assisted-1 work may be classified for JAVA, assisted-2 work may be classified for XML, assisted-3 work may be classified C code, etc.

**[0035]** Additionally, support is provided by the integrated assist system for limiting the extent to which assisted work may be run on a standard processor. These business rules are implemented using system control structure 500. For example, crossover flag field 502 is used to specify whether assisted work may be run on a standard processor. If crossover flag field 502 is set to 'N', then assisted work may not be run on a standard processor as long as any assist processors are online. Alternatively, it may be determined that assisted work is to be executed on a standard processor only when the standard processor has no other standard work to do. This rule may be implemented by setting crossover flag 502 to ON and honorpriority flag 504 to OFF.

**[0036]** When there is standard work to do but there is also assisted work with a higher priority than the standard work, then both crossover flag 502 and honorpriority flag 504 would be set to ON. When there is standard work to do and the assisted work has higher priority than the standard work, but the standard work has a higher priority than a set priority bar, then crossover flag 502 and honorpriority flag 504 would both be set to ON and the priority bar field 510 would be set to a priority threshold.

**[0037]** Support is provided for limiting the extent to which assisted work is run on a standard processor. The following scenarios are provided in order to describe the business rules provided by the dispatch component.

1. Never – in this case CrossOver Flag 502 is off indicating that no crossover is to occur.

2. When the standard processor has no standard work to do – in this case, CrossOver flag 502 is on and HonorPriority flag 504 is off.

3. When there is standard work to do but there is assisted work with higher priority than the standard work – in this case, CrossOver flag 502 is on and HonorPriority flag 504 is on.

4. When there is standard work to do and the assisted work has higher priority than the standard work but the standard work has higher priority than a "priority bar" – in this case, CrossOver flag 502 is on, HonorPriority flag 504 is on and the PriorityBar 510 is set.

**[0038]** For case 1, a standard processor looks only at work maintained on the standard processor control structure 402 (or, in the single control structure case, deals only with work that is "standard"). An assist-n processor looks only at work maintained on the assist-n processor control structure 404 (or, in the single control structure case, deals with work that is "assisted-n").

**[0039]** For case 2, a standard processor looks for work maintained on the standard processor control structure 402 (or, in the single control structure case, deals with work that is "standard"), and upon not finding any such eligible work looks for work maintained on the assist processor control structures 404 (or, in the single control structure case, looks for work that is "assisted"). An assist-n processor looks only at work maintained on the assist-n processor control structure 404 (or, in the single control structure case, deals only with work that is "assisted-n").

**[0040]** For case 3, a standard processor looks for the highest priority 304 eligible work maintained across all of the processor types. It may do this by keeping a single control structure in priority order, or may keep separate control structures for each processor type and search them all. There are many ways to accomplish such a search of all structures looking for the highest priority work. A simple way is to search each in turn for its highest priority work and remember both the element with the highest priority and the work type structure in which that element was located. An alternate is to treat the structures as being in some particular order, locating the highest priority eligible work in

the "first", then if the "next" control structure has work higher than the work located in the "previous" control structure then search that, and continue until all control structures have been examined. An assist-n processor looks only at work maintained on the assist-n processor control structure 404 (or, in the single control structure case, deals only with work that is "assisted-n"). Another method of search that may be used is described in U.S. Patent application entitled "Method, System, and Storage Medium for Prioritizing Work Elements Across Multiple Queues", referenced above. The process returns to step 202.

**[0041]** Case 4 is similar to case 3, but the processing done on a standard processor looks at assist-n structures 404 only when the priority work located on the standard processor structure is of priority 304 below a "priority bar". An example of this is:

Priority work on standard processor is priority 50

Priority work on assist processor is priority 75

Priority Bar is 40

In this case, the standard processor work is used, even though the assist processor has a higher priority balue.

A second example is:

Priority work on standard processor queue 402 is priority 30

Priority work on assist processor queque 404 is priority 75

Priority Bar is 40

In this case, the assit processor work is used, because the standard processor has a lower priority value than the bar.

A third example is:

Priority work on standard processor queque 402 is priority 30

Priority work on assist processor is queque 404 is priority 20

Pariority Bar is 40

In this case, the standard processor work is used.

**[0042]** The integrated assist system further includes utilization reporting component. It is generally known that a variety of monitoring products exist that run on an operating system and provide utilization information such as the level of engagement (e.g., how busy) the system is, or system capacity information, etc. For example, IBM's® Resource Measurement Facility (RMF) monitors system resources and reports back to the operating system. The integrated assist system includes a DIAGNOSE 204 instruction that returns utilization information to operating system 119 relating to standard and assist processors that has been accumulated by hypervisor 112. DIAGNOSE 204 includes subcode that new or updated monitoring programs may use to get the integrated assist logical processor information integrated into the standard logical processor information while providing a segregated view of operations information for each type of processor back to the operating system 119.

**[0043]** In a multi-tier processing environment, the integrated assist processor is incorporated into the common or standard processor and memory attachment infrastructure of the database server in a manner that eliminates the need for a physically separate application server tier within the network. By providing such a heterogeneous processor environment for both application and database programs, the application assist processor can provide enhanced functional benefits by executing the application program on the typically more reliable multi-function database servers within the network which, in turn, allows the application programs to benefit from the advantages of improved qualities of service, improved server reliability, and improved server scalability that contemporary large scale database servers typically provide.

**[0044]** By executing both the application programs and their associated database programs on a common application/database server platform, both the application

programs and the database programs can interact with each other and directly share data with each other, without requiring any physical network "wires" and their associated communications programming stacks (e.g. TCP/IP, firewalls, etc.) that would otherwise be required to interconnect physically separate application and database servers. Such integrated "inter-program" communications and direct memory based data sharing capabilities can significantly reduce the overall complexity of the network, increase the total networks reliability, and can significantly reduce the network's transaction processing latencies.

**[0045]** The integrated assist system integrates an application assist processor into a database server platform to form a heterogeneous processor complex capable of executing both the application and associated database programming in a manner that is both more efficient and reliable than 3-tier networks and which can compete with such networks on a cost of ownership basis. The integrated assist system further integrates the application assist processor into the common processor and memory attachment infrastructure of the database server that eliminates the need for a physically separate application server tier within the network. By providing such a heterogeneous processor environment for both application and database programs, such servers can provide enhanced functional benefits by executing the application program on the typically more reliable multi-function database servers within the network which, in turn, allow the application programs to benefit from the advantages of improved qualities of service, improved server reliability, and improved server scalability that contemporary large scale database servers typically provide.

**[0046]** As described above, the present invention can be embodied in the form of computer-implemented processes and apparatuses for practicing those processes. The present invention can also be embodied in the form of computer program code containing instructions embodied in tangible media, such as floppy diskettes, CD-ROMs, hard drives, or any other computer-readable storage medium, wherein, when the computer program code is loaded into and executed by a computer, the computer becomes an

apparatus for practicing the invention. The present invention can also be embodied in the form of computer program code, for example, whether stored in a storage medium, loaded into and/or executed by a computer, or transmitted over some transmission medium, such as over electrical wiring or cabling, through fiber optics, or via electromagnetic radiation, wherein, when the computer program code is loaded into and executed by a computer, the computer becomes an apparatus for practicing the invention. When implemented on a general-purpose microprocessor, the computer program code segments configure the microprocessor to create specific logic circuits.

**[0047]** While the invention has been described with reference to exemplary embodiments, it will be understood by those skilled in the art that various changes may be made and equivalents may be substituted for elements thereof without departing from the scope of the invention. In addition, many modifications may be made to adapt a particular situation or material to the teachings of the invention without departing from the essential scope thereof. Therefore, it is intended that the invention not be limited to the particular embodiments disclosed for carrying out this invention, but that the invention will include all embodiments falling within the scope of the claims.